SIMPLE Working Group
Internet-Draft
Expires: April 25, 2003

B. Campbell
J. Rosenberg
R. Sparks
dynamicsoft
P. Kyzivat
Cisco Systems

October 25, 2002

Instant Message Sessions in SIMPLE
draft-campbell-simple-im-sessions-01

Abstract

The SIP MESSAGE method is used to send instant messages, where each message is independent of any other message. This is often called pager-mode messaging, due to the fact that this model is similar to that of most two-way pager devices. Another model is called session-mode. In session-mode, the instant messages are part of a media session that provides ordering, a security context, and other functions. This media session is established using a SIP INVITE, just as an audio or video session would be established.

This document describes a The Message Session Relay Protocol (MSRP), a mechanism for transmitting session-mode messages with minimal relay support. Additionally, this document describes using the SDP offer/ answer model to initiate such sessions.

Table of Contents

1. Introduction

The MESSAGE [7] extension to SIP [2] allows SIP to be used to transmit instant messages. Instant messages sent using the MESSAGE method are normally independent of each other. This approach is often called pager-mode messaging, since it follows a model similar to that used by many two-way pager devices. Pager-mode messaging makes sense for instant message exchanges where a small number of messages occur.

There are also applications in which it is useful for instant messages to be associated together in some way. For example, a user may wish to join a text conference, participate in the conference for some period of time, then leave the conference. This usage is analogous to regular media sessions that are typically initiated, managed, and terminated using SIP. We commonly refer to this model as session-mode messaging.

One of the primary purposes of SIP is the management of media sessions. Session-mode messaging can be thought of as a media session like any other. This document describes a method to use SIP to manage message sessions. This document does not propose an actual message session mechanism; there may any number of mechanisms that are appropriate for different applications and environments.

This document describes the motivations for session-mode messaging, the Message Session Relay Protocol, and the use of the SDP offer/answer mechanism for initiating MSRP session.

2. Motivation for Session-mode Messaging

Message sessions offer several advantages over pager-mode messages. For message exchanges that include more than a small number of message transactions, message sessions offer a way to remove messaging load from intervening SIP proxies. For example, a minimal session setup and teardown requires one INVITE/ACK transaction, and one BYE transaction, for a total of 5 SIP messages. Normal SIP request routing allows for all but the initial INVITE transaction to bypass any intervening proxies that do not specifically request to be in the path for future requests. Session-mode messages never cross the SIP proxies themselves, unless proxies also act as message relays.

Each pager mode message involves a complete SIP transaction, that is, a request and a response. Any pager-mode message exchange that involves more than 2 or 3 MESSAGE requests will generate more SIP requests than a minimal session initiation sequence. Since MESSAGE is normally used outside of a SIP dialog, these requests will typically traverse the entire proxy network between the endpoints.

Due to network congestion concerns, the MESSAGE method has significant limitations in message size, a prohibition against overlapping requests, etc. Much of this has been required because of perceived limitations in the congestion-avoidance features of SIP itself. Work is in progress to mitigate these concerns.

However, session-mode messages are always sent over a reliable, congestion-safe transport. Therefore, there are no restrictions on message sizes. There is no requirement to wait for acknowledgement, so that message transactions can be overlapped.

Message sessions allow greater efficiency for secure message exchanges. The SIP MESSAGE request inherits the S/MIME features of SIP, allowing a message to be signed and/or encrypted. However, this approach requires public key operations for each message. With session-mode messaging, a session key can be established at the time of session initiation. This key can be used to protect each message that is part of the session. This requires only symmetric key operations, and no additional certificate exchanges are required after the initial exchange. The establishment of the session key is done using standard techniques that apply to voice and video, in addition to instant messaging.

Finally, SIP devices can treat message sessions like any other media sessions. Any SIP feature that can be applied to other sorts of media sessions can equally apply to message sessions. For example, conferencing [9], third party call control [10], call transfer [11], QoS integration [12], and privacy [13] can all be applied to message sessions.

Messaging sessions can also reduce the overhead in each individual message. In pager-mode, each message needs to include all of the SIP headers that are mandated by RFC 3261 [2]. However, many of these headers are not needed once a context is established for exchanging messages. As a result, messaging session mechanisms can be designed with significantly less overhead.

3. Scope of this Document

This document describes the use of MSRP between endpoints, or via one or two relays, where endpoints have advance knowledge of the relays. It does not provide a mechanism for endpoints to determine whether a relay is needed, or for endpoints to discover the presence of relays.

4. Protocol Overview

The Message Session Relay Protocol (MSRP) provides a mechanism for transporting session-mode messages between endpoints. MSRP also

contains primitives to allow the use of one or two relay devices. MSRP uses connection oriented, reliable network transport protocols only. It is intrinsically NAT and firewall friendly, and allows network connections to be shared across multiple message sessions. MSRP allows participants to positively associate message sessions with specific connections, and does not depend upon connection source address, which may be obscured by NATs.

MSRP uses the following primitives:

SEND: Used to actually send message content from one endpoint to another.

VISIT (LEAVE): Used by an endpoint to establish a session association to the opposite endpoint, or to a relay that was selected by the opposite endpoint. "LEAVE" is used to remove the association.

BIND(RELEASE): Used by an endpoint to establish a session at a relay, and allow the opposite endpoint to visit that relay. "RELEASE" is used to remove the session.

The simplest use case for MSRP is a session that goes directly between endpoints, with no intermediaries involved. Assume A is an endpoint that wishes to establish a message session, and B is the endpoint invited by A. A invites B to participate in a message session by sending B two URIs. The first URI represents a the destination to which B should send messages. The second represents a identifier that B can use to connect to , or visit, A, and to which A will send messages. Both URIs are temporary, and must not duplicate the local or remote URIs used in any other active sessions.

B "visits" A by connecting to A and sending a VISIT request containing the URI that A provided it. This establishes state at A that associates the connection from B with the URI. B then responds to the invitation, informing A that B accepts the URI that A assigned to it. A can now send messages to B using the SEND method with B's URI as the target. Likewise, B can invoke the SEND method targeting A's URI.

When either party wishes to end the session, it informs the other party with a SIP BYE. B may end the message session by sending a "LEAVE" request, telling A that B will no longer use the URI. If no other sessions were using the network connection, B closes the connection. alternately, A can end the session by simply invalidating state associating the URIs and connections, and dropping the connection if no more sessions are using it.

This creates a race condition between A and B when closing a session. It seems that this does not really cause a problem, in that if an attempt to close a session fails because the other party closed it first, the desired outcome is still achieved.

The end to end case looks something like the following. (Note that the example shows a logical flow only; syntax will come later in this document.)

```
A->B (SDP): offer (i-am:a@a, you-be:b@a)

B->A (MSRP): VISIT (b@a)

A->B (MSRP): 200 OK

B->A (SDP): answer(i-am b@a)

A->B (MSRP): SEND (b@a)

B->A (MSRP): 200 OK

B->A (MSRP): SEND (a@a)

A->B (MSRP): 200 OK

B->A (MSRP):  LEAVE (b@a)

A->B (MSRP): 200 OK
```

A slightly more complicated case involves a single relay, known about in advanced by one of the parties. The endpoint with the pre-existing relationship with the relay uses the BIND method to establish session state in the relay. The relay returns a pair of temporary URIs, one for local use, and one to offer to the remote endpoint. For endpoints A and B, and relay R, the flow would look like the following:

```
A->R: MSRP: BIND

R->A: MSRP: 200 OK (local:a@r, remote:z@r)

A->B (SDP): offer (i-am:a@r, you-be:z@r)

B->R (MSRP): VISIT (z@r)

R->B (MSRP): 200 OK
```

```
B->A (SDP): answer(i-am:b@r)

A->R (MSRP): SEND (z@r)

R->B (MSRP): SEND (z@r)

B->R (MSRP): 200 OK

R->A (MSRP): 200 OK

B->R (MSRP): SEND (a@r)

R->A (MSRP): SEND (a@r)

A->R (MSRP): 200 OK

R->B (MSRP): 200 OK

A->R (MSRP): RELEASE(a@r)

B->R (MSRP): LEAVE(z@r)
```

5. SDP Offer-Answer Exchanges for MSRP Sessions.

MSRP sessions will typically be initiated using the Session Description Protocol (SDP) [1] offer-answer mechanism, carried in SIP [2] or any other protocol supporting it.

5.1 Use of the SDP M-line

The SDP m-line takes the following form:

```
m=<media> <port> <protocol> <format list>
```

For non-RTP media sessions, The media field specifies the top level MIME media type for the session. For MSRP sessions, the media field MUST have the value of "message". The proto field MUST designate the message session mechanism and transport protocol, separated by a "/" character. For MSRP, left part of this value MUST be "msrp". For example, "msrp/tcp" for MSRP over TCP, "msrp/tls" for MSRP over TLS, etc. The format list MUST indicate the MIME content-types that the endpoint is willing to accept in the payload of SEND requests. If any of the allowed types are compound in nature, that is, they allow one or more arbitrary MIME body parts to be imbedded within them, then the format list MUST include the content-types allowed for the imbedded parts.

The following example illustrates an m-line for a CPIM message session, where the endpoint is willing to accept payloads of plain text or HTML, which may appear at the top level of the payload, or may be imbedded inside a message/cpim body part.

```
   m=message 49232 msrp/tcp message/cpim text/plain text/html
```

5.2 URI Negotiations

MSRP offer/answer negotiations require the transmissions of one or two URIs. The first identifies the party sending the offer or answer. the second offers a temporary URI for the opposite party to use to connect to the sending party, or to a relay operating on behalf of the sending party. These URIs are respectively designated as "i-am" and "you-be". The i-am URI MUST be present in each SDP offer and answer. The you-be URI MUST be present if the party sending the offer or answer is proposing that its peer connect to the sending party or such a relay, and MUST NOT be present otherwise.

The URI parameters are represented in SDP as a combination of domain part in the c-line address parameter and the user parts in a-line attributes. The "msrp:" scheme is implied, and MUST not be present in the a-line. For example, the following indicates an i-am URI of "msrp:2s93i@example.com" and a you-be URI of "msrp:8djese@example.com

```
        c=IN IP4 example.com
        m=message 7394 msrp/tcp text/plain
        a=i-am:2s93i
        a=you-be:8djese
```

   This approach implies that the i-am and you-b URIs will always share the same domain part. Is there any scenario where this would not be desired?

Both the i-am and the you-b URIs MUST be temporary URIs assigned just for this particular session. They MUST NOT duplicate any URI in use in any other session hosted by the endpoint. Further, since the peer endpoint may use the you-be URI to identify itself when connecting, it should be hard to guess, and protected from eavesdroppers. This will be discussed in more detail in the Security Consideration section.

5.3 Example SDP Exchange

Endpoint A wishes to invite Endpoint B to a MRSP session. A offers the following session description containing the following lines:

```
        c=IN IP4 alice.example.com
        m=message 7394 msrp/tcp message/cpim text/plain
        a=i-am:2s93i9
        a=you-be:8r90ew
```

Endpoint B chooses to participate, opens a TCP connection to alice.example.com:7394, and successfully performs a VISIT transaction passing the URI of "msrp:8r90ew@alice.example.com". B indicates that it has accomplished this by answering with:

```
        c=IN IP4 alice.example.com
        m=message 7394 msrp/tcp message/cpim text/plain text/html
        a=i-am:8r90ew
```

A may now send IMs to B by executing SEND transactions targeted to "msrp:8r90ew@alice.example.com", and B may SEND to "msrp:2s93i9@alice.example.com"

5.4 Session Hosting

MSRP requires the use of connection oriented transport protocols. Part of the offer answer process involves determining which endpoint will "host" the session, that is, which will listen for the network connection. When a relay is involved, the hosting endpoint is the one that initializes session state at the relay with a BIND request. An endpoint indicates its desire to host a session by including a "you-be" URI in its SDP.

Typically, the offerer will also be the host. If the offerer is in a position to accept a connection, or has access to a relay that can accept a connection on its behalf, then it SHOULD offer to host the session. If the offerer does in fact include a you-be URI, the answerer SHOULD accept the offerer's desire to host by visiting the URI provided, and, if successful, returning the offered you-be URI in the i-am URI of the answer.

There may be situations where the endpoint inviting a peer to participate in an MDRP cannot, or does not wish to, host the connection. For example that endpoint may be behind a NAT or firewall that prevents inbound connections. In this case, the initiating endpoint MAY send an SDP offer with no you-be URI. If the answerer receives such an offer, it SHOULD act as session host, and return a you-be URI in the answer. Of course, if neither party is able and willing to host the session, then they cannot participate in an MSRP session.

This probably requires the offerer to accept the offered you-be URI in its ACK. However, we are trying to specify this in terms of

the SDP offer/answer model only, and not depend on SIP's three phase INVITE. We could simply have the inviting party not include an offer, but in this case the invitee is likely to assume an RTP session is desired, unless some other mechanism is invoked to communicate the desired session type.

There may be scenarios where the invitee is required by policy to use a particular relay, and by implication, must act as host on all session, even if the inviter included the you-be URI in the offer. The invitee may decline to accept the inviter's proposal to host by returning a different you-be URI, and an i-am URI that does not match the you-be URI of the offer. In this situation, the inviter SHOULD allow the invitee to host.

Finally, there may be scenarios where both parties are required by policy to use different relays. This is similar to the previous case where the invitee overrides the inviter's proposal to host, except that the inviter does not return the you-be URI that the invitee proposed as its i-am URI in the ACK request. Each endpoint BINDs with its relay, and neither performs a VISIT. When one of the endpoints attempts a SEND operation, its relay will attempt to connect to the peer's relay. This will be described in more detail in the Relay Behavior section.

6. The Message Session Relay Protocol

The Message Session Relay Protocol (MSRP) is a text based, messaging oriented protocol for the transfer of instant messages in the context of a session. MSRP uses the UTF8 character set.

MSRP messages MUST be sent over reliable, congestion-controlled, connection-oriented transport protocols, such as TCP.

6.1 MSRP messages

MSRP messages are either requests or responses. Requests and responses are distinguished from one another by the first line. The first line of a Request takes the form of the request-start entry below. Likewise, the first line of a response takes the form of response-start. The syntax for an MSRP message is as follows:

```
       msrp-message = request-start/response-start
                    *(header CRLF)
                     [CRLF body]
       request-start = "MSRP" SP length SP Method CRLF
       response-start= "MSRP" SP length SP Status-Code SP Reason CRLF
       length = 1*DIGIT  ; the length of the message
       Method = SEND / BIND / RELEASE / VISIT /LEAVE
       header = Client-Authenticate / Server-Challenge /
                 Transaction-ID / Target-URI/ Content-Type
                Local-URI / Remote-URI
       Status-Code = 200    ;Success
                   / 400    ;Bad Request
                   / 401    ;Authentication Required
                   / 403    ;Forbidden
                   / 481    ;No session
                   / 500    ;Cannot Deliver
                   / 506    ;duplicate session
       Reason = token ; Human readable text describing status
       msrp-uri= msrp HCOLON user "@" domain
       Client-Authenticate = "CAuth" HCOLON username "," nonce
                             "," digest-response
       Server-Challenge = "SChal" HCOLON nonce
       Transaction-ID = "TR-ID" HCOLON token
       Content-Type = "Content-Type" HCOLON quoted-string
       Target-URI = "T-URI" HCOLON msrp-uri
       Local-URI = "L-URI" HCOLON msrp-uri
       Remote-URI = "R-URI" HCOLON msrp-uri
```

This syntax is not complete. In particular, we will need more work on Server-Challenge and Client-Authenticate header fields.

All requests and responses MUST contain at least a T-URI and a TR-ID header field. Messages MAY contain other fields, depending on the method or response code.

6.2 MSRP Transactions

An MSRP transaction consists of exactly one request and one response. A response matches a transaction if it share the same TR-ID value, and if the T-URI value in the response matches a the URI for the endpoint that sent the request.

BIND and VISIT transactions are always hop by hop. However, SEND transactions are end to end, meaning that under normal circumstances the response is sent by the peer endpoint, even if there are one or more intervening relays.

Endpoints MUST select TR-ID header field values in requests so that they are not repeated by the same endpoint in scope of the given session. The TR-ID space of each endpoint is independent of that of its peer. Endpoints MUST NOT infer any semantics from the TR-ID header field beyond what is stated above. In particular, TR-ID values are not required to follow any sequence.

MSRP Transactions complete when a response is received, or after a timeout interval expires with no response. Endpoints MUST treat such timeouts in exactly the same way they would treat a 500 response. The size of the timeout interval is a matter of local policy.

6.3 MSRP Sessions

AN MSRP session is a context in which a series of IMs are sent, using SEND requests. A session has two endpoints (a host and a visitor) and may have one or more relays. A session is identified by the tuple of the host and visitor URIs. Note that an endpoint that is involved in multiple sessions will likely have completely different URIs in one session than in another.

6.4 Initiating an MSRP session

When an endpoint wishes to engage a peer endpoint in a message session, it invites the peer to communicate using an SDP offer, carried over SIP or some other protocol supporting the SDP offer/answer model. For the purpose of this document, we will refer to the endpoint choosing to initiate communication as the inviter, and the peer being invited as the invitee.

The inviter SHOULD act as host for the session. An endpoint is said to host a session if one of two conditions are true. The host either directly listens for a connection from the peer endpoint, and maintains session state itself, or it initialized session state at a relay that will listen for a connection from the peer, using a BIND request. The peer that is not the host is designated as the visitor. The inviter MAY allow the invitee to act as host, if it is prevented from accepting connections by network topology or policy, and is not able to bind to a relay to act on its behalf.

If the inviter chooses to host the session directly, that is without using a relay, it MUST perform the following steps:

1. Construct a local and visitor MSRP URI . These MUST share the same domain part, which MUST be resolvable to the inviter. The user parts MUST be distinct from each other. The user part of each URI SHOULD be temporary, SHOULD be hard to guess, and MUST not duplicate the any URI used in other active sessions hosted by

the inviter.

2. Listen for a connection from the peer.

3. Construct an SDP offer as described in Section 5, including the list of allowed IM payload formats in the format list. The inviter maps the local URI to the i-am attribute, and the visitor URI to the you-be attribute, as described in Section 5.2

4. Send the SDP offer using the normal processing for the signaling protocol.

If the inviter chooses to ask the invitee to host the session, it MUST perform the following steps instead:

1. Construct a host URI as described above. It does not construct a visitor URI. Since the inviter will not accept network connections, it is not important that the URI resolve to the inviter.

2. Construct an SDP offer as described above. The M-line port value can be any non-zero value, since it will not actually be used.

3. Send the offer using normal processing for the signaling protocol.

When the invitee receives the SDP offer and chooses to participate in the session, it must choose whether of act as the host or the visitor. A you-be attribute in the offer indicates that the inviter wishes to host, in which case the invitee SHOULD act as the visitor.

If the invitee chooses to participate as a visitor, it MUST perform the following steps:

1. Connect to the host address and port from the C-line and M-line in the SDP offer, using the transport protocol from the M-line.

2. Construct a VISIT request, which MUST contain the following information:

    1. A T-URI header field containing the visitor URI.

    2. A TR-ID header field containing a unique transaction ID.

    3. A size field containing the overall size of the message.

3. Send the request and wait for a response

4. If the transaction succeeds, send a SDP answer via the signaling protocol, according to the following rules:

    1. The C-line is copied unmodified from the offer.

    2. The M-Line contains the port from the original offer and protocol fields from the original offer, and a format list describing the SEND payload media types that the invitee is willing to accept.

    3. No you-be attribute

    4. A i-am attribute containing the user part of the visitor URI, that is, the same value that was in the you-be attribute on the offer.

The invitee MAY choose to attempt to host the session under some circumstances. If the SDP offer did not contain a you-be attribute, the invitee MUST host the session, otherwise it is not possible to participate in the session at all. If the offer did contain a you-be attribute, but the attempt to connect to the host failed, the invitee MAY attempt to host the session. Otherwise, the invitee MAY attempt to take over as host because of a local policy requiring it.

If the invitee chooses to host the session, it MUST perform the following steps:

1. Construct a local and visitor MSRP URI . These MUST share the same domain part, which MUST be resolvable to the invitee. The user parts MUST be distinct from each other. Both URI SHOULD be temporary, SHOULD be hard to guess, and MUST not duplicate URIs used in any active sessions hosted by the invitee.

2. Listen for a connection from the peer.

3. Construct an SDP answer as described in Section 5, including the list of allowed IM payload formats in the format list. The invitee maps the local URI to the i-am attribute, and the visitor URI to the you-be attribute, as described in Section 5.2

4. Send the SDP offer using the normal processing for the signaling protocol.

When the inviter receives the SDP answer, it must determine who will continue to host the session. If the offer contained a you-be attribute and the answer did not, the inviter MUST continue host the session. If the offer did not contain a you-be attribute, and the answer did, then the inviter MUST allow the invitee to host the

session. If both the offer and the answer contained you-be attributes, this indicates that the invitee is attempting to become host. The inviter SHOULD allow the inviter to continue as host.

If the inviter chooses to allow the invitee to host the session, it MUST send an acknowledgment of the SDP answer. If the inviter chooses to host in spite of the invitee's attemt to take over, the acknowlegement must repeat the i-am attribute from the original offer.

If the inviter chooses not to host, it must perform the following steps:

1. Release any resources it acquired in expectation of hosting the session, if any.

2. Connect to the host address and port from the C-line and M-line in the SDP answer, using the transport protocol from the M-line.

3. Construct a VISIT request, which MUST contain the following information:

    1. A T-URI header field containing the visitor URI from the SDP answer.

    2. A TR-ID header field containing a randomly selected initial transaction ID.

    3. A size field containing the overall size of the message.

4. Send the request and wait for a response

5. If the VISIT succeeds, acknowledge the answer via the signaling protocol. If either the connection attempt or the VISIT transaction fail, acknowledge the answer, then initiate the tear-down of the request using the signaling protocol.

6.5 Handling VISIT requests

An MSRP endpoint that is hosting a session will receive a VISIT request from the visiting endpoint. When an endpoint receives a VISIT request, it MUST perform the following procedures:

1. Check if state exists for a session using a visitor URI that matches the T-URI of the VISIT request. If so, and no previous VISIT request for that URI has occurred, then return a 200 response, and save state designating the connection on which the

request was received as the visitor leg of the session.

2. If the session exists, but a VISIT request has already been received for it, return a 506 response, and do not change session state in any way.

3. If no matching session exists, return a 481 request, and do not change session state in any way.

6.6 Sending Instant Messages on a Session

Once a MSRP session has been established, either endpoint may send instant messages to its peer using the SEND method. When an endpoint wishes to do so, it MUST construct a SEND request according to the following process:

1. Set the T-URI header field to the peer URI. (If the sender is the host, it used the visitor URI, and vice versa.)

2. Insert the message payload in the body, and the media type in the Content-Type header field. The media type MUST match one of the types in the format list in the SDP provided by the peer.

3. Set the TR-ID header field to a unique value

4. Send the request on the connection associated with the session.

5. If a 200 response code is received, the transaction was successful.

6. If a 500 response code is received, the transaction failed, but may possibly be successful if retried. The endpoint MAY retry the request with a new TR-ID header field value. If the endpoint receives 500 responses more than a threshold number of times in a row, it should assume the session has failed, and initiate teardown via the signaling protocol. The threshold value is a matter of local policy.

7. If any other response code is received, the endpoint SHOULD assume the session has failed, and initiate teardown.

When an endpoint receives a SEND request, it MUST perform the following steps.

1. Verify that the T-URI header field value matches the local URI for an existing session.

2. If it does, render the message to the end user, and return a 200 response. The definition of "render" is a matter of local policy.

3. If it does not match an existing session, return a 481 response.

6.6.1 Ending a Session

When either endpoint in an MSRP session wishes to end the session, it first signals its intent using the normal processing for the signaling protocol. For example, in SIP, it would send a BYE request to the peer. After agreeing to end the session, each endpoint MUST release any resources acquired as part of the session. The process for this differs depending on whether the endpoint is the host or the visitor.

The host MUST destroy local state for the session. This involves completely removing the state entry for this session, invalidating both the local and remote URIs. If the host is using an MSRP relay, it MUST send a RELEASE request to the relay. It MUST send the RELEASE on the same connection used for the BIND. The RELEASE MUST include the local URI in the T-URI header field.

The visitor MUST send a LEAVE request on the same connection on which it sent the original VISIT. The LEAVE request MUST include the visitor URI in the T-URI header field. The visitor MUST then invalidate any local state for the session.

> This approach creates a race condition in that both the RELEASE and LEAVE requests cause the session state to be invalidated at the host. Whichever of these occurs second will receive a 481 response. We may wish to consider if both of these methods are necessary. Is it sufficient to leave state cleanup entirely to the host?

6.7 Managing Session State and Connections

A MSRP session is represented by state at the host device. As mention previously, session state is identified by the tuple formed by the local and remote URIs. And active session also has a connection associated with each of these URIs. The connection associated with the remote URI is known as the visiting connection. The connection associated with the local URI is known as the host connection. Note that when the session state is hosted directly by an endpoint, the host connection may not involve a physical network connection; rather it is a logical connection the device maintains with itself.

A given network connection may be associated with more than one session. In fact, when the session state is hosted by a relay, it is entirely possible for a single connection to be the host connection for one session, and the visiting connection for another.

When session state is destroyed for any reason, the hosting device SHOULD check to see if each connection is currently associated with any other sessions. If not, the device SHOULD drop the connection.

If a connection fails for any reason, the session hosting device MUST invalidate the session state. This is true regardless of whether the dropped connection is the host or visiting connection. Once a connection is dropped, the associated session state MUST NOT be reused. If the endpoints wish to continue to communicate after a connection failure, they must initiate a new session.

6.8 MSRP Relays

6.8.1 Establishing Session State at a Relay

An endpoint that wishes to host a MSRP session MAY do so by initiating session state at a MSRP relay, rather than hosting directly. An endpoint may wish to do this because network topology or local policy prevents a peer from connecting directly to the endpoint. The use of a relay should not be the default case, that is, a hosting endpoint that is not prevented from doing so by topology or policy SHOULD host the session directly. In order to use a relay, an MSRP endpoint MUST have knowledge of that endpoints existence and location.

We previously mentioned how an endpoint wishing to host a MSRP session constructs a local and remote URI. When using a relay, the endpoint delegates that responsibility to the relay.

To establish session state at a relay, the endpoint MUST perform the following steps:

1. Construct a BIND request with a T-URI that refers to the relay.

2. Open a network connection to the relay at the relays address and the well-known port for MSRP relays, or at another port if so configured.

3. Send the BIND request on the connection.

4. Respond to any authentication request from the relay.

5. If the response has a 200 status code, use the URI in the L-URI

header field as the local URI, and the URI in the R-URI header field as the remote URI. The endpoint uses these URIs in exactly the same manner as it had constructed them itself.

A MSRP relay listens for connections to its well-known port at all times. When it receives a BIND request, it SHOULD authenticate the request, either using digest-authentication, TLS authentication, or some other authentication mechanism. If authentication succeeds, the relay performs the following steps:

1. Verify the client is authorized to BIND to this relay. If not, return a 403 response and make no state change.

2. If the client is authorized, construct a local and remote MSRP URI. The domain part of both URIs MUST be the same, and MUST resolve to the relay. The URIs SHOULD be temporary, and hard to guess. The URIs MUST not duplicate URIs used in any active sessions hosted by the relay. If the relay wishes the visiting endpoint to connect over a point other than the MSRP relay well-know port, it MAY add the port number to visitor URI.

3. Create state for the session. The relay MUST associate the connection on which it received the BIND request with the local URI.

4. Return a 200 response, with the T-URI header field value copied from the request, the local URI in the L-URI header field, and the remote URI in the R-URI field.

When an MSRP relay receives a VISIT request, it MUST perform the following steps:

1. Check the T-URI header field value to see it matches the remote URI for an existing session state entry.

2. If not, return a 481 response and make no state changes

3. If it matches, but a connection has already been associated with the remote URI, then return a 506 response and make no state changes.

4. If it matches, and no connection has been associated with the remote URI, then the VISIT succeeds. The relay associates the connection on it received the VISIT request with the remote URI for the session, and returns a 200 response.

6.8.2 Removing Session State from a relay

An MSRP relay SHOULD remove state for a session when any of the following conditions occur:

- o The host performs sends an UNBIND request with a T-URI that matches the local-URI associated with the session.
- o The visitor sends a LEAVE request with a T-URI that matches the remote-URI associated with the session.
- o Either the host or visitor network connection is reset by the peer, or fails for any reason.

6.8.3 Sending IMs across an MSRP relay

Once a session is established at a relay, the host and visitor may exchange IMs by sending SEND requests. Under normal circumstances, the relay does not respond to SEND requests in any way. If the T-URI of the SEND request matches the peer URI for the session, the relay MUST forward the request unchanged. Likewise, if the relay receives a response, where the T-URI matches the peer URI for the session, it MUST forward the request unchanged.

If the relay receives a SEND request that does not match the opposite URI of the session, the relay MUST return a 404 response. If a SEND request arrives on a connection that has no associated sessions, the relay MUST return a 481 response. Note that this may occur in a situation where the relay has removed session state because of a failure in the peer connection.

6.8.4 Relay Pairs

In rare circumstances, two relays may be required in a session. For example, two endpoints may exist in separate administrative domains, where each domain's policy insist that all sessions must cross that domain's relay. In this scenario, each endpoint BINDs with its own endpoint, and believes itself to be the session host. Neither endpoint sends a VISIT request.

For a relay to support this scenario, it must implement some special behavior under the following circumstances:

- o A session is in a half-complete state; that is, the relay has returned a local and remote URI in a response to a BIND request, but has not observed a VISIT request using the remote URI.

o A SEND request arrives on a connection that is associated with at least one such "half complete" sessions, and the T-URI refers to a domain for which the relay is not responsible.

Under these circumstances, the relay SHOULD attempt to open a connection with a device responsible for the domain in the T-URI, and forward the request. If a connection already exists, it SHOULD reuse the connection. Likewise, if a relay receives a SEND request on a connection not associated with a session, and the T-URI is associated with an existing connection, it SHOULD forward the request on that connection. If there is no such associated connection, it SHOULD return a 500 response.

This scenario will often result in two effectively one-way connections between a pair of relays. Additionally, if one of the relays is behind a NAT, this scenario may fail. We believe this is acceptable for the two relay case, which should be fairly rare.

6.9 Method Descriptions

This section summarizes the purpose of each MSRP method. All MSRP requests MUST contain the T-URI and TR-ID header fields. All requests MUST contain a length field in the start line that indicates the overall length of the request, including any body. Additional requirements exist depending on the individual method. Except where otherwise noted, all requests are hop by hop.

6.9.1 BIND

The BIND method is used by a host endpoint to establish session state at a relay. BIND requests SHOULD be authenticated. BIND requests MAY contain the CAuth header fields.

A successful response to a BIND request MUST contain the L-URI and R-URI header fields.

6.9.2 LEAVE

The LEAVE method is used by a visiting endpoint to request that the host invalidate a session. The T-URI field MUST match that of the corresponding VISIT request.

6.9.3 RELEASE

The RELEASE method is used by a host endpoint to request a relay to invalidate a session. The T-URI header field in a RELEASE request MUST match the L-URI header field returned in the successful response

to the corresponding BIND request.

6.9.4 SEND

The SEND method is used by both the host and visitor endpoints to send instant messages to its peer endpoint. The T-URI header field MUST contain the temporary URI for the peer, that is a message sent to the visitor MUST use the visitor URI, and a message sent to the host MUST use the host URI. SEND requests SHOULD contain a MIME body part. The body MUST be of a media type included in the format list in the SDP provided by the peer. If a body is present, the request MUST contain a Content-Type header field identifying the media type of the body.

> There has been discussion that we could make media type identification more efficient by using the numeric index of the media type as listed in the SDP format list, rather than the full media type. This would have an advantage of space efficiency. The editor holds the opinion that the ease of readability more than offsets any space inefficiency, and is more in keeping with the philosophy that led us to choose a text-based protocol in the first place. The editor will, however, yield to any contrary consensus on the matter.

Unlike other methods, SEND requests are end to end in nature. This means the request is consumed only by the opposite endpoint. Any intervening relays merely forward the request on towards its target.

6.9.5 VISIT

The visiting endpoint uses the VISIT method to associate a network connection with the session state at the hosting device, which could be either the host endpoint or a relay operating on behalf of the host endpoint. The T-URI header field MUST match the visitor URI associated with the session.

> Notice that there is no authentication operation for the VISIT request. This is because the visitor URI acts as a shared secret between host and the visitor. This puts certain requirements on the handling of the visitor URIs that are discussed in Section 9

6.10 Response Code Descriptions

This section summarizes the various response codes. Except where noted, all responses MUST contain the T-URI header field containing the URI of the endpoint that is the intended consumer of the response. Responses are never consumed by relays.

All responses MUST include the TR-ID header, with a value matching that of the corresponding request.

6.10.1 200

The 200 response code indicates a successful transaction.

6.10.2 400

A 400 response indicates a request was unintelligible.

6.10.3 401

A 401 response indicates authentication is required. 401 responses MUST NOT be used in response to any method other than BIND. A 401 response MUST contain a SChal header field.

6.10.4 403

A 403 response indicates the authenticated identity of the user attempting a BIND request is not authorized to perform such an action. A 403 response SHOULD NOT be used in response to any method other than BIND.

6.10.5 481

A 481 response indicates that no session exists that is associated with the connection on which the request arrived, and has a peer URI that matches the T-URI of the corresponding request.

6.10.6 500

A 500 response indicates that a relay was unable to deliver a SEND request to the target. A 500 response SHOULD NOT be used except for when multiple relays exist in a session, and a relay is unable to deliver the request to the next relay. Endpoints SHOULD NOT return 500 responses. 500 responses MUST NOT be returned in response to any method other than SEND.

6.10.7 506

A 506 response indicates that a VISIT request occurred in which the T-URI indicates a session that is already associated with another connection. A 506 response MUST NOT be returned in response to any method other than VISIT.

6.11 Header Field Descriptions

This section summarizes the various header fields. MSRP header fields are single valued; that is, they MUST NOT occur more than once in a particular request or response.

6.12 T-URI

The T-URI Header field indicates the target of a request or a response. The semantics of T-URI vary with the method in question.

6.13 TR-ID

The TR-ID header field contains a transaction identifier used to map a response to the corresponding request. A TR-ID value MUST be unique among all values used by a given endpoint inside a given session. MSRP elements MUST NOT assume any additional semantics for TR-ID.

6.14 CAuth

The CAuth header field is used by a host endpoint to respond to offer digest authentication credentials to a relay, usually in response to a digest-authentication challenge. CAuth MUST NOT be present in a request of any method other than BIND.

6.15 SChal

The SChal header field is used by a relay to carry the challenge in a digest authentication attempt. The SCHal header MUST NOT be used in any message except for a 401 response.

The semantics of digest authentication in the context of MSRP are not fully defined at the time of this writing. We expect this work to be completed in a future version of this document, at which time the syntax of CAuth and SChal may change.

6.16 Content-Type

The Content-Type header field is used to indicate the MIME media type of the body. Content-Type MUST be present if a body is present.

6.17 L-URI

The L-URI header field is used by a relay to indicate the local URI for a session created with a BIND request. The L-URI header field MUST be present in a 200 response to a BIND request, and MUST NOT be present in any other request or response.

6.18 R-URI

The R-URI header field is used by a relay to indicate the remote URI for a session created with a BIND request. The L-URI header field MUST be present in a 200 response to a BIND request, and MUST NOT be present in any other request or response.

7. Examples

This section shows some example message flows for various common scenarios. The examples assume SIP is used to transport the SDP exchange. Details of the SIP messages and SIP proxy infrastructure are omitted for the sake of brevity. In the examples, assume the inviter is sip:alice@atlanta.com and the invitee is sip:bob@biloxi.com. In any given MSRP message, an "xx" in the length field indicates the actual length of the message

7.1 No Relay

1. Alice constructs a local uri of msrp:iau39@alicepc.atlanta.com and a remote uri of msrp: 9342iw@alicepc.atlanta.com, and listens for a connection on TCP port 7777.

2. Alice->Bob (SIP): INVITE sip:bob@biloxi.com

```
c=IN IP4 alicepc.atlanta.com
m=message 7777 msrp/tls text/plain
a=i-am:iau39
a=you-be:9342iw
```

3. Bob->Alice: Open TCP connection to Alicepc.atlanta.com:7777, and perform TLS handshake.

4. Bob->Alice (MSRP):

```
MSRP xx VISIT
T-URI: 9342iw@alicepc.atlanta.com
TR-ID: msrp:sie09s
```

5. Alice->Bob (MSRP):

```
MSRP xx 200 OK
T-URI: 9342iw@alicepc.atlanta.com
TR-ID: sie09s
```

6. Bob->Alice (SIP): 200 OK

```
c=IN IP4 alicepc.atlanta.com
m=message 7777 msrp/tls text/plain
a=i-am:9324iw
```

7. Alice->Bob (SIP): ACK

8. Alice->Bob (MSRP):

```
MSRP xx SEND
T-URI: msrp:9234iw@alicepc.atlanta.com
TR-ID: 123
Content-Type: "text/plain"
Hi, I'm Alice!
```

9. Bob->Alice (MSRP):

```
MSRP xx 200 OK
T-URI: msrp:iau39@alicepc.atlanta.com
TR-ID: 123
```

10. Bob->Alice (MSRP):

```
MSRP xx SEND
T-URI: msrp:iau39@alicepc.atlanta.com
TR-ID: 456
Content-Type: "text/plain"

Hi, Alice! I'm Bob!
```

11. Alice->Bob (MSRP):

```
MSRP xx 200 OK
T-URI: msrp:9234iw@alicepc.atlanta.com
TR-ID: 456
```

12. Alice->Bob (SIP): BYE

13. Alice invalidates session

14. Bob->Alice (MSRP):

```
MSRP xx LEAVE
T-URI: msrp:9342iw@alicepc.atlanta.com
TR-ID: 987
```

15. Alice->Bob (MSRP):

```
MSRP xx 481 No Session T-URI: 9342iw@alicepc.atlanta.com
TR-ID: 987
```

16. Bob invalidates local state for the session.

17. Bob->Alice (SIP): 200 OK

7.2 Single Relay

This scenario introduces an MSRP relay at relay.atlanta.com.

1. Alice->Relay (MSRP): Alice opens a TLS connection to the relay, and sends the following:

```
MSRP xx BIND
T-URI: msrp:relay.atlanta.com
TR-ID: 321
```

2. Relay->Alice (MSRP):

```
MSRP xx 200 OK
T-URI: msrp:relay.atlanta.com
TR-ID: 321
L-URI: msrp:iau39@relay.atlanta.com
R-URI: msrp:9342iw@relay.atlanta.com:7777
```

3. Alice->Bob (SIP): INVITE sip:bob@biloxi.com

```
c=IN IP4 relay.atlanta.com
m=message 7777 msrp/tls text/plain
a=i-am:iau39
a=you-be:9342iw
```

4. Bob->Alice: Open TCP connection to relay.atlanta.com:7777, and perform TLS handshake.

5.   Bob->Relay (MSRP):

```
MSRP xx VISIT
T-URI: 9342iw@relay.atlanta.com
TR-ID: msrp:sie09s
```

6.   Relay->Bob (MSRP):

```
MSRP xx 200 OK
T-URI: 9342iw@relay.atlanta.com
TR-ID: sie09s
```

7.   Bob->Alice (SIP): 200 OK

```
c=IN IP4 relay.atlanta.com
m=message 7777 msrp/tls text/plain
a=i-am:9324iw
```

8.   Alice->Bob (SIP): ACK

9.   Alice->Relay (MSRP):

```
MSRP xx SEND
T-URI: msrp:9234iw@relay.atlanta.com
TR-ID: 123
Content-Type: "text/plain"
Hi, I'm Alice!
```

10. Relay->Bob (MSRP):

```
MSRP xx SEND
T-URI: msrp:9234iw@relay.atlanta.com
TR-ID: 123
Content-Type: "text/plain"
Hi, I'm Alice!
```

11. Bob->Relay (MSRP):

```
MSRP xx 200 OK
T-URI: msrp:iau39@relay.atlanta.com
TR-ID: 123
```

12. Relay->Alice (MSRP):

```
MSRP xx 200 OK
T-URI: msrp:iau39@relay.atlanta.com
TR-ID: 123
```

13. Bob->Relay (MSRP):

```
MSRP xx SEND
T-URI: msrp:iau39@relay.atlanta.com
TR-ID: 456
Content-Type: "text/plain"

Hi, Alice! I'm Bob!
```

14. Relay->Alice (MSRP):

```
MSRP xx SEND
T-URI: msrp:iau39@relay.atlanta.com
TR-ID: 456
Content-Type: "text/plain"

Hi, Alice! I'm Bob!
```

15. Alice->relay (MSRP):

```
MSRP xx 200 OK
T-URI: msrp:9234iw@relay.atlanta.com
TR-ID: 456
```

16. Relay->Bob (MSRP):

```
MSRP xx 200 OK
T-URI: msrp:9234iw@relay.atlanta.com
TR-ID: 456
```

17. Alice->Bob (SIP): BYE

18. Alice->Relay (MSRP):

```
MSRP xx RELEASE T-URI: msrp:iau39@relay.atlanta.com
TR-ID: 42
```

19. Relay->Alice (MSRP): (relay invalidates session state)

```
MSRP xx 200 OK
T-URI: msrp:iau39@relay.atlanta.com
TR-ID: 42
```

20. Bob->Relay (MSRP):

```
MSRP xx LEAVE
T-URI: msrp:9342iw@relay.atlanta.com
TR-ID: 987
```

21. Alice->Bob (MSRP):

```
MSRP xx 481 No Session T-URI: msrp:9342iw@relay.atlanta.com
TR-ID: 987
```

22. Bob invalidates local state for the session.

23. Bob->Alice (SIP): 200 OK

7.3 Two Relays

In this scenario, both Alice and Bob are required by local policy to route all sessions through a local relay.

1. Alice->AtlantaRelay (MSRP): Alice opens a TLS connection to the relay, and sends the following:

```
MSRP xx BIND
T-URI: msrp:relay.atlanta.com
TR-ID: 321
```

2. AtlantaRelay->Alice (MSRP):

```
MSRP xx 200 OK
T-URI: msrp:relay.atlanta.com
TR-ID: 321
L-URI: msrp:iau39@relay.atlanta.com
R-URI: msrp:9342iw@relay.atlanta.com:7777
```

3. Alice->Bob (SIP): INVITE sip:bob@biloxi.com

```
   c=IN IP4 relay.atlanta.com
   m=message 7777 msrp/tls text/plain
   a=i-am:iau39
   a=you-be:9342iw
```

4.   Bob determines that, due to local policy, he must host the session through his own proxy.

5.   Bob->BiloxiRelay (MSRP): Bob opens a TLS connection to his relay, and sends the following:

```
   MSRP xx BIND
   T-URI: msrp:relay.biloxi.com
   TR-ID: 934
```

6.   BiloxiRelay->Bob(MSRP):

```
   MSRP xx 200 OK
   T-URI: msrp:relay.biloxi.com
   TR-ID: 934
   L-URI: msrp:oeksd@relay.biloxi.com
   R-URI: msrp:kdowsw@relay.biloxi.com:7777
```

7.   Bob->Alice (SIP): 200 OK

```
   c=IN IP4 relay.biloxi.com
   m=message 8910 msrp/tls text/plain
   a=i-am:oeksd
   a=you-be: kdowsw
```

8.   Alice sees that Bob has requested to host instead. However, local policy requires her to insist on using the atlanta relay.

9.   Alice->Bob (SIP): ACK

```
   c=IN IP4 relay.atlanta.com
   m=message 7777 msrp/tls text/plain
   a=i-am:iau39
```

10. Alice->AtlantaRelay (MSRP):

```
   MSRP xx SEND
   T-URI: msrp:oeksd@relay.biloxi.com
   TR-ID: 123
   Content-Type: "text/plain"
```

```
      Hi, I'm Alice!
```

11. AtlantaRelay recognizes it has received a SEND request targeted to a foreign domain. AtlantaRelay does a DNS resolution to find the relay at relay.biloxi.com, opens a connection to the well know port, and forwards the request.

12. BiloxiRelay->Bob (MSRP):

```
      MSRP xx SEND
      T-URI: msrp:oeksd@relay.biloxi.com
      TR-ID: 123
      Content-Type: "text/plain"
      Hi, I'm Alice!
```

13. Bob->BiloxiRelay (MSRP):

```
      MSRP xx 200 OK
      T-URI: msrp:iau39@relay.atlanta.com
      TR-ID: 123
```

14. BiloxiRelay recognizes it has received a response targeted to a foreign domain. BiloxiRelay does a DNS resolution to find the relay at relay.atlanta.com, opens a connection to the well know port, and forwards the request. Note that BiloxiDomain cannot positively determine that it already has a connection _from_ relay.atlanta.com, as a NAT may obscure the source address on the original connection.

15. AtlantaRelay->Alice (MSRP):

```
      MSRP xx 200 OK
      T-URI: msrp:iau39@relay.atlanta.com
      TR-ID: 123
```

16.
   Alice->Bob (SIP): BYE

17.
   Alice->AtlantaRelay (MSRP):

```
      MSRP xx RELEASE T-URI: msrp:iau39@relay.atlanta.com
      TR-ID: 42
```

18. Relay->Alice (MSRP): (relay invalidates session state)

```
MSRP xx 200 OK
T-URI: msrp:iau39@relay.atlanta.com
TR-ID: 42
```

19. Bob->BiloxiRelay (MSRP):

```
MSRP xx RELEASE
T-URI: msrp:oeksd@relay.biloxi.com
TR-ID: 938
```

20. BiloxiRelay->Bob (MSRP): (relay invalidates session state)

```
MSRP xx 200 OK
T-URI: msrp:oeksd@relay.biloxi.com
TR-ID: 42
```

21. Bob->Alice (SIP): 200 OK

22. After a locally configured timeout period with no activity, AtlantaRelay and BiloxiRelay are free to drop their respective connections to each other.

8. Changes introduced in 01 version

Version 01 is a significant re-write. References to COMEDIA were removed, as it was determined that COMEDIA would not allow connections to be used bidirectionally in the presence of NATs. Significantly more discussion of a concrete mechanism has been added to make up for no longer using COMEDIA. Additionally, this draft and draft-campbell-cpimmsg-sessions (which would have also changed drastically) have now been combined into this single draft.

9. Security Considerations

There are a number of security considerations for MSRP, some of which are mentioned elsewhere in this document. This section discusses those further, and introduces some new ones.

9.1 Sensitivity of the Remote URI

The remote URI of a MSRP session is used by the visiting endpoint to identify itself to the hosting device, regardless of whether the

session is directly hosted by the host endpoint, or is hosted by a relay. If an attacker were able to aquire the remote-URI, either by guessing it or by evedropping, there is a window of opportunity in which the attacker could hijack the session by sending a VISIT request to the host device before the true visiting endpoint. Because of this sensitivity, the remote URI SHOULD be constructed in a way to make it difficult to guess, and should be sufficiently random so that it is unlikely to be reused. All mechanisms used to transport the remote URI to the visitor and back to the host MUST be protected from eavesdroppers and man-in-the-middle attacks. This includes the VISIT request itself, as well as SDP exchange mechanism.

Therefore an MSRP device MUST support the use of TLS for at least the VISIT request, which by extension indicates the endpoint MUST support the use of TLS for all MSRP messages. Further, MSRP connections SHOULD actually be protected with TLS. Further, an MSRP endpoint MUST be capable of using the security features of the signaling protocol in order to protect the SDP exchange and SHOULD actually use them on all such exchanges. End-to-end protection schemes SHOULD be preferred over hop-to-hop schemes for protection of the SDP exchange.

9.2 End to End Protection of IMs

Instant messages can contain very sensitive information. As a result, as specified in RFC 2779 [4], instant messaging protocols need to provide for encryption, integrity and authentication of instant messages. Therefore MSRP endpoints MUST support the end-to-end encryption and integrity of bodies sent via SEND requests, using the session key generation mechanism described in MIKEY [5].

9.3 CPIM compatibility

MSRP sessions may be gatewayed to other CPIM [14]compatible protocols. If this occurs, the gateway MUST maintain session state, and MUST translate between the MSRP session semantics and CPIM semantics that do not include a concept of sessions. Furthermore, when one endpoint of the session is a CPIM gateway, instant messages SHOULD be wrapped in "message/cpim" [6] bodies. Such a gateway MUST include "message/cpim" as the first entry in its SDP format list. MSRP endpoints sending instant messages to a peer that has included 'message/cpim" as the first entry in the format list SHOULD encapsulate all instant message bodies in "message/cpim" wrappers. All MSRP endpoints SHOULD support the S/MIME features of that format.

10. IANA Considerations

MSRP will likely require the IANA registration of an well-known port for MSRP relays. Additionally, we may have to register the "i-am" and

"you-be" attributes of the MSRP SDP exchange.

11. Open Issues

This specification is far from complete, particularly in the areas of error handling and security considerations. If the working group chooses to fully specify MSRP, additional work is required in those areas.

Further specification of the MSRP URI scheme will also be required. For example, this version does not describe URI comparison rules, or rules for resolving a URI using the DNS. The concept of an "msrps" URI scheme that required the use of TLS on all hops was discussed in the design team. This concept is not documented in this version, as it is not fully enough developed to include. Such a scheme is likely to appear in a future version of this document.

12. Contributors

The following people contributed substantially to this ongoing effort:

Rohan Mahy
Allison Mankin
Jon Peterson
Brian Rosen
Dean Willis

Normative References

[1] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Campbell, B., "Instant Message Sessions using the CPIM Message Format.", draft-campbell-simple-cpimmsg-sessions-00.txt (work in progress), October 2002.

[4] Day, M., Aggarwal, S. and J. Vincent, "Instant Messaging / Presence Protocol Requirements", RFC 2779, February 2000.

[5] Arkko, J., "MIKEY: Multimedia Internet KEYing", draft-ietf-msec-mikey-04 (work in progress), August 2002.

[6] Atkins, D. and G. Klyne, "Common Presence and Instant Messaging

Message Format", draft-ietf-impp-cpim-msgfmt-06 (work in progress), February 2001.

Informational References

[7] Campbell, B. and J. Rosenberg, "Session Initiation Protocol Extension for Instant Messaging", RFC 3428, September 2002.

[8] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[9] Rosenberg, J. and H. Schulzrinne, "Models for Multi Party Conferencing in SIP", draft-ietf-sipping-conferencing-models-01 (work in progress), July 2002.

[10] Rosenberg, J., Peterson, J., Schulzrinne, H. and G. Camarillo, "Best Current Practices for Third Party Call Control in the Session Initiation Protocol", draft-ietf-sipping-3pcc-02 (work in progress), June 2002.

[11] Sparks, R., "SIP Call Control - Transfer", draft-ietf-sip-cc-transfer-05 (work in progress), July 2001.

[12] Camarillo, G., Marshall, W. and J. Rosenberg, "Integration of Resource Management and Session Initiation Protocol (SIP)", RFC 3312, October 2002.

[13] Peterson, J., "A Privacy Mechanism for the Session Initiation Protocol (SIP)", draft-peterson-sip-privacy-longterm-00 (work in progress), March 2002.

[14] Crocker, D., Diacakis, A., Mazzoldi, F., Huitema, C., Klyne, G., Rose, M., Rosenberg, J., Sparks, R., Sugano, H. and J. Peterson, "A Common Profile for Instant Messaging (CPIM)", draft-ietf-impp-cpim-03 (work in progress), August 2002.

Authors' Addresses

Ben Campbell
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75024

EMail: bcampbell@dynamicsoft.com

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936

EMail: jdrosen@dynamicsoft.com

Robert Sparks
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75024

EMail: rsparks@dynamicsoft.com

Paul Kyzivat
Cisco Systems
Mail Stop LWL3/12/2
900 Chelmsford St.
Lowell, MA 01851

EMail: pkzivat@cisco.com

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.